Yale University
Department of Computer Science

# Relative Knowledge and Belief
(EXTENDED ABSTRACT)

Michael J. Fischer Lenore D. Zuck

YALEU/DCS/TR-589
December 1987

This work was supported in part by the National Science Foundation under grant DCR-8405478 and by the Office of Naval Research under Contract N00014-82-K-0154.

# Relative Knowledge and Belief

(Extended Abstract)

Michael J. Fischer Lenore D. Zuck

**Abstract**

Motivated by recent research in cryptographic protocols and formal theories of knowledge, we present a logic of feasible and probabilistic knowledge. Our notion of *relative knowledge* captures the idea of feasibly computable knowledge, and our notion of *relative belief* captures the idea of feasibly computable knowledge with a degree of confidence $\alpha < 1$. We illustrate the power of our definitions by characterizing the state of knowledge of the verifier after running an interactive proof of knowledge of a square root in $\mathbf{Z}_n^*$.

## 1 Introduction

Much research in distributed computing and cryptographic protocols has centered on solving problems of multi-agent systems with various elements of *uncertainty*, e.g., the Byzantine generals problem (where uncertainty stems from the possible faultiness of the processes), mutual exclusion (where uncertainty stems from the asynchrony of the system), mental poker, etc. Intuitively, uncertainty implies lack of knowledge, and overcoming it implies establishing some degree of knowledge. This has led to the natural observation that a useful way to analyze distributed and cryptographic systems is in terms of knowledge and how communication changes the processors' state of knowledge [CM86].

Our goal is to formalize the concepts of knowledge needed for reasoning about zero-knowledge interactive proofs of language membership [GMR85] and of knowledge [GHY85, FFS87,TW87]. Knowledge arises there in three places:

1. The knowledge the prover wishes to convey to the verifier (in the case of a proof of knowledge as opposed to a proof of language membership).
2. The knowledge the verifier gains after having run the protocol.
3. The the knowledge the verifier does *not* gain after having run the protocol.

The need for such formalization is apparent. Feige, Fiat and Shamir [FFS87] say, "The notion of 'knowledge' is very fuzzy, and a-priori it is not clear what proofs of knowledge actually prove." They discuss the difficulties of obtaining an adequate definition of knowledge; indeed, their formal definition of an interactive proof system of knowledge makes

This work was supported in part by the National Science Foundation under grant DCR-8405478 and by the Office of Naval Research under Contract N00014-82-K-0154.

no explicit reference to knowledge concepts. Tompa and Woll [TW87] propose a different formal definition of an interactive proof system of knowledge; they also make no explicit reference to knowledge concepts.

In this paper, we develop a formal framework for knowledge that is adequate for expressing (1) and (2). This allows us, for example, to express the soundness property for an interactive proof system [TW87]. (Cf. Our Theorem 2.) Our framework is not yet adequate to express the zero-knowledge property, which asserts that (3) includes all "important" facts. Defining a formal system of knowledge and showing that no knowledge property expressible in that system is unintentionally conveyed by the protocol is not enough; one must also demonstrate that the formal system is sufficiently expressive, i.e. able to express all "important" properties, perhaps by showing that any protocol that is zero knowledge in this formal sense is also zero knowledge in the sense of [GMR85]. We leave to future work the problem of extending our knowledge framework to handle (3).

The formal concept of knowledge in computer science has been an active area of research for the past several years, see, e.g., [HM84,FI86]. The main contribution of these works is the formalization of what we call *implicit knowledge*. Intuitively, a fact $\psi$ is implicit knowledge for agent $i$ if it is necessarily true based on $i$'s local view of the world. In other words, $i$ *knows* $\psi$, written $\mathsf{K}_i\psi$, if $\psi$ is true in all states of the world that look the same to $i$ as the present state. This definition of knowledge satisfies the *knowledge axiom*

$$\mathsf{K}_i\psi \supset \psi$$

which says that $i$ only knows true statements.

As many people have observed, implicit knowledge is inadequate for reasoning about the knowledge appearing in interactive proof systems. The reasons are manyfold:

a. Implicit knowledge (and, in fact, all the notions of formal knowledge that we are aware of) deals with knowledge of predicates. However, in an interactive proof system the prover may want to convince the verifier it knows some value in a prescribed set of possibile values, e.g., one of the four square roots of $y$ modulo a number $n$ that is the product of two large primes. (We denote this set by $\sqrt{y} \bmod n$.) We therefore need a concept of knowledge that captures *knowledge of multi-valued functions*.[1]

b. Implicit knowledge ignores the computational complexity of extracting knowledge from the local view. For example, since $\sqrt{y} \bmod n$ is uniquely determined by $y$ and $n$, then the prover always implicitly knows it. This implicit knowledge is however not feasible since the prover might not be able to obtain any element in $\sqrt{y} \bmod n$ efficiently (in probabilistic polynomial time), i.e., this knowledge is not *feasible*. The point of an interactive proof of knowledge of $\sqrt{y} \bmod n$ is that the prover be able to efficiently produce some element in the set.

c. Following our previous example, since we believe that finding any element in $\sqrt{y} \bmod n$ is computationally difficult, it seems that the prover can never know $\sqrt{y} \bmod n$ in a feasible way. Suppose, however, that the prover has a secret tape $s$. The prover can

[1]The obvious approach of defining knowledge of $f(x)$ as a conjunction of knowledge of each bit of $f(x)$ fails on two grounds: It does not extend to multi-valued functions, for knowing the $i^{\text{th}}$ bit of one of the possible values of $f(x)$ for each $i$ does not imply knowing all of the bits of the *same* value. It also does not extend to the probabilistic case, for knowing each bit of $f(x)$ with high confidence does not imply knowing the actual value of $f(x)$ with similar confidence.

compute $s^2$ mod $n$. If, by chance, it equals $y$ then the prover knows an element of $\sqrt{y}$ mod $n$. This may indeed be the essence of the prover's knowledge in an interactive proof system, not that it can compute the square root of an arbitrary quadratic residue $y$, but only that it sometimes happens, for whatever reason, to already know such a square root. Our knowledge system must therefore be able to deal with such *accidental knowledge*.

d. Implicit knowledge lacks the expressive power which is required when reasoning about probabilistic protocols (of which interactive proofs are an example). In such a system, an agent might, e.g., "know" that $\psi$ is true in 99% of the worlds that look the same to it as the current world. However, if $\psi$ is false in the remaining 1% of the worlds, $\psi$ is not *always* true and hence $i$ does implicitly know $\psi$.

e. Not only are interactive proof systems explicitly probabilistic, they are implicitly nondeterministic. For example, every cheating prover defines a different system when interacting with a correct verifier. Each of these systems is probabilistic, yet the verifier cannot tell which system it takes part in, nor is it realistic to assume a probability distribution on the possible protocols used by the cheating prover. Its knoweldge at the end of the protocol must account for this nondetreminism.

To deal with (a), we extend implicit knowledge to deal directly with multi-valued functions of the state, and we observe that knowledge of a predicate can be treated as a special case of knowledge of a multi-valued function.

To deal with (b) and (c), we present a logic of "relative" knowledge that allows us to express both feasible knowledge and accidental knowledge. The idea behind our logic is that there be an efficient algorithm $M$ which, given the local view of an agent $i$ as input, either outputs one of the values of the function $f$ at the current state or says '?' (meaning "I don't know"). We call such an $M$ an *$i$-feasible knowledge generator for $f$* and say that *$i$ knows $f$* in a state *relative* to $M$ if $M$ produces a value other than '?' given the agent's current local view.

When applied to a predicate $\psi$, the knowledge generator $M$ either outputs **true** or '?', and it outputs **true** only when $\psi$ really is true. Thus, we can view $M$ as a verification procedure for $\psi$—it "proves" $\psi$ holds by outputting **true**. The statement "$i$ knows $\psi$ relative to $M$" then means that "$M$ verifies for $i$ that $\psi$ holds". Since $M$ depends only on $i$'s local view, if it verifies $\psi$ in state $g$, then it also verifies $\psi$ in all states $g'$ that look the same to $i$ as $g$. Hence, relative knowledge of $\psi$ implies implicit knowledge of $\psi$.

Relative knowledge gives us a whole range of degrees of knowledge, each depending on the properties of the particular knowledge generator $M$. At the one extreme, a knowledge generator that never says '?' is a feasible algorithm that an agent can use to find some value of the function at every global state. At the other extreme, a knowledge generator that always says '?' is trivially correct but gives no useful information.

To deal with (d), we present a logic of probabilistic knowledge. Probabilistic knowledge often plays the same role as true knowledge, but, because the knowledge axiom does not hold, we refer to such knowledge as *belief*.[2] We introduce *implicit belief* which corresponds to knowledge in probabilistic systems. Unlike knowledge, belief is not absolute but is associated with a degree of confidence.

[2]Our notion of belief is of a probabilistic nature. This is different from other notions of belief based on the failure of the knowledge axiom for other reasons [FH85].

We then combine the definitions of implicit belief and relative knowledge to obtain *relative belief* by allowing the knowledge generator some probability of error. We call such unreliable knowledge generators *belief generators*. Not only does relative belief fail to satisfy the knowledge axiom, but it is also non-monotonic; extra information can lower one's degree of confidence.

Finally, to deal with (e), we introduce non-determinism. We assume each agent has a set of probabilistic protocols from which it (non-deterministically) chooses one to execute. While each agent's set of possible protocols is commonly known to all the other agents, the chosen protocol is not. The tuple of protocols collectively chosen by the agents constitutes a probabilistic system. A formula is known or believed if it is known or believed in all such tuples.

We demonstrate the utility of these definitions by applying them to a particular zero-knowledge interactive proof that the prover knows an element in $\sqrt{y} \bmod n$ [GMR85,TW87]. We give a succinct and rigorous expression and proof of the soundness condition for that protocol. This suggests that we have achieved at least partial success in combining the classical approach to the formal theory of knowledge with the notions of knowledge that have appeared in modern cryptographic protocols.

## Related Work

Moses [Mos87] addresses problem (b) by adding to implicit knowledge the requirement that there be an efficient algorithm for deciding whether or not $\mathsf{K}_i\psi$ holds, given only $i$'s local view. This notion of feasible knowledge can be expressed within our framework as knowledge of the characteristic function $\chi_\psi$ of the predicate $\psi$ relative to a knowledge generator $M$ that never outputs '?'. (The characteristic function $\chi_\psi(g) = \{\mathsf{true}\}$ if $\psi$ is true at $g$, and $\chi_\psi(g) = \{\mathsf{false}\}$ if $\psi$ is false at $g$.) This same $M$ also decides whether the implicit knowledge formula $\mathsf{K}_i\psi$ holds, so it satisfies Moses's requirement.

Moses's definition appears to be quite reasonable in the context of knowledge-based protocols [HF85,HZ87], where decisions must be made on the basis of an agent's knowledge. However, it is too restrictive for our purposes, for by requiring that it capture implicit knowledge exactly, it fails to account for (c). In the same work, Moses acknowledges that in order to deal with issues in cryptography, his framework must be extended to talk directly about functions instead of only predicates and to include probability (our problems (a) and (d)).

In [Hal87], Halpern showed a way of defining "probabilistic knowledge" (which appears to be similar to our *implicit belief*) and claimed it can be easily extended to allow formalizing the knowledge of a verifier after an interactive proof; he gave an example of what such a formal statement might look like.

## 2 The Computational Model

We consider terminating distributed systems with a set $\mathcal{A}$ of participating agents. Formal definition of a similar system appears in, e.g., [Hal86]; we briefly sketch it here.

The set of *global states* of a system $\mathcal{R}$ is denoted by $\mathcal{G} = \mathcal{G}_{\mathcal{R}}$. The set of *local states* of each agent (process) is denoted by $\mathcal{V}$. For every agent $i \in \mathcal{A}$, we assume some function

$\nu_i: \mathcal{G} \to \mathcal{V}$ that maps each $g \in \mathcal{G}$ to $i$'s local *view* of $g$. Given two global states $g$ and $g'$, we say that $g$ and $g'$ are indistinguishable to $i$, denoted by $g \sim_i g'$, if $i$ has the same local view in both, i.e., if $\nu_i(g) = \nu_i(g')$.

We assume each agent $i \in \mathcal{A}$ runs a protocol which is a polynomial time Turing machine. The protocols together with an initial global state define the possible legal runs of the system, where each run is a finite sequence of global states. We therefore identify a *system* with its set of runs. A system $\mathcal{R}$ is *deterministic (probabilistic)* if the underlying protocols are deterministic (probabilistic).

We will generally be interested in probabilistic systems. We can consider a probabilistic system to be a deterministic system in which each participant $i$ has an additional (sufficiently long) *random tape* which it can read during the course of the computation. Assuming each cell of each random tape is chosen uniformly and independently from $\{0, 1\}$, one gets an induced probability distribution on runs in the natural way.

For technical convinience, we assume that agents do not forget, i.e., in every run $r$, for every $k$, $1 \leq k \leq |r|$, $\nu_i(r_k)$ includes $\nu_i(r_{k-1})$, where $r_\ell$ is the $\ell^{\text{th}}$ state in $r$.

## 3 Knowledge

In this section, we define implicit and relative knowledge for deterministic protocols.

### 3.1 Implicit Knowledge

Let $\mathcal{R}$ be a deterministic system with global states $\mathcal{G} = \mathcal{G}_{\mathcal{R}}$. We assume a set of *base facts* (predicates) on the global states that varies from application to application. For example, if each global state $g$ includes a number $n(g) \in \mathrm{N}$, then we might consider a base fact *prime* such that for every $g \in \mathcal{G}$,

$$\textit{prime}(g) \qquad \text{iff} \qquad n(g) \text{ is a prime number.}$$

Similarly, we assume a set of *base multi-valued functions* with domain $\mathcal{G}$ that varies from application to application. For example, we can add a base function *prime-factors*: $\mathcal{G} \to 2^{\mathrm{N}}$ by defining

$$\textit{prime-factors}(g) = \{p \mid p \text{ is a prime factor of } n(g)\}.$$

We define a set of *facts* and a set of *functions* over $\mathcal{G}$ inductively from the base functions and facts.

1. Base facts are facts.
2. If $f$ is a function, then $\mathsf{K}_i f$ is a fact for every $i \in P$.
3. If $\psi$ is a fact, then $\mathsf{K}_i \psi$ is a fact for every $i \in P$.
4. If $\psi$ and $\xi$ are facts, then so are $\neg\psi$ and $\psi \vee \xi$.
5. Base functions are functions.
6. If $\psi$ is a fact, then $f_\psi$ is a function.

$f_\psi$ is a multi-valued function associated with the fact $\psi$. It simplifies our subsequent definitions by allowing us to define knowledge of facts in terms of knowledge of functions.

The following inductively defines the semantics of facts and functions. The semantics of a fact is defined as a satisfiability relation $\models_\mathcal{R}$ between a global state and the fact. Each function is defined as a mapping that takes a global state to a (possibly empty) set of values.

1. $g \models_\mathcal{R} p$ iff $p(g)$, where $p$ is a base fact.
2. $g \models_\mathcal{R} \mathsf{K}_i f$ iff $\bigcap\{f(g') \mid g' \sim_i g\} \neq \emptyset$, where $f$ is a function.
3. $g \models_\mathcal{R} \mathsf{K}_i \psi$ iff $g \models_\mathcal{R} \mathsf{K}_i f_\psi$, where $\psi$ is a fact.

4a. $g \models_\mathcal{R} \neg\psi$ iff $g \not\models_\mathcal{R} \psi$.

4b. $g \models_\mathcal{R} \psi \vee \xi$ iff $g \models_\mathcal{R} \psi$ or $g \models_\mathcal{R} \xi$.

5. The value of a base function is assumed to be known.
6. $f_\psi(g) = \{\mathsf{true}\}$ if $g \models_\mathcal{R} \psi$, and $f_\psi = \emptyset$ (the empty set) if $g \models_\mathcal{R} \neg\psi$, where $\psi$ is a fact.

**Remarks**

- Here and in the sequel, we omit mention of the system $\mathcal{R}$ when it is clear from context.
- If $f$ is a function then $g \models \mathsf{K}_i f$ means "$i$ knows some value of $f$ at the current global state".
- If $\psi$ is a fact, then $g \models \mathsf{K}_i \psi$ means "$i$ knows that $\psi$ is true at the current global state".
- If for every $g$ the value of $f$ depends only on $\nu_i(g)$, then for every $g$, $g \models \mathsf{K}_i f$ iff $f(g) \neq \emptyset$.
- If $\psi$ is a fact then $g \models \mathsf{K}_i \mathsf{K}_j \psi$ implies that $g \models \mathsf{K}_i \psi$. However, this is not true for functions.

## 3.2 Relative Knowledge

Implicit knowledge ignores the computational problem of determining from the local view of agent $i$ when a formula $\mathsf{K}_i \psi$ holds. For example, assuming $n$ and *prime-factors* are as defined previously, then $g \models \mathsf{K}_i(\textit{prime-factors})$ always holds, for the prime factors of an integer $n$ are uniquely determined by $n = n(g)$. On the other hand, the problem of factoring $n$ is believed to be computationally difficult, so there is no known efficient algorithm for computing *prime-factors*$(g)$, and hence $i$ has no feasible way of finding the prime factors of $n$ that she implicitly knows. Here we are interested in defining knowledge of a fact or function where computational limitations *are* taken into account.

Let $i$ be an agent in the system, let $f$ be a function with domain $\mathcal{G}$, and let $M$ be a probabilistic polynomial time Turing machine. $M$ is an *$i$-feasible knowledge generator for $f$* if for every $g \in \mathcal{G}$, $M$ on input $\nu_i(g)$ outputs an element in $f(g) \cup \{\text{'?'}\}$.

We are now in a position to define relative knowledge. We say that agent $i$ *knows a function $f$ in state $g \in \mathcal{G}$ relative to a machine $M$*, denoted by

$$g \models_\mathcal{R} \mathsf{K}_i^M f,$$

iff $M$ is an $i$-feasible knowledge generator for $f$ and $M(\nu_i(g)) \neq$ '?'. Similarly, we say that agent $i$ *knows a fact* $\psi$ *in state* $g \in \mathcal{G}$ *relative to a machine* $M$, denoted by

$$g \models_{\mathcal{R}} \mathsf{K}_i^M \psi,$$

iff $g \models_{\mathcal{R}} \mathsf{K}_i^M f_\psi$.

Relative knowledge implies implicit knowledge, i.e., $\mathsf{K}_i^M f$ implies $\mathsf{K}_i f$. This is because $g \models \mathsf{K}_i^M f$ implies that $M$ is an $i$-feasible knowledge generator for $f$ and $M(\nu_i(g)) \in f(g)$. It follows that for every state $g' \sim_i g$, '?' $\neq M(\nu_i(g)) = M(\nu_i(g')) \in f(g')$, so $M(\nu_i(g)) \in \bigcap\{f(g') \mid g' \sim_i g\}$. Thus $g \models \mathsf{K}_i f$.

## 3.3 Probabilistic Relative Knowledge

The notion of relative knowledge generalizes easily to allow the knowledge generator itself to be a probabilistic algorithm. Such an algorithm $M$ computes a random function in which a probability is associated with each possible output of $M(v)$. We say that $M$ is *i-feasible* for $f$ if for every $g \in \mathcal{G}$, every output of $M$ on input $\nu_i(g)$ with non-zero probability is in $f(g) \cup \{$'?'$\}$.

To generalize the notion of relative knowledge to probabilistic knowledge generators, we add a confidence value $\alpha$ to the knowledge operator and say that in state $g \in \mathcal{G}$ *agent* $i$ *knows a function* $f$ *with confidence* $\alpha$ *relative to a machine* $M$, denoted by

$$g \models_{\mathcal{R}} \mathsf{K}_i^{(\alpha,M)} f,$$

iff $M$ is an $i$-feasible probabilistic knowledge generator for $f$ and

$$\text{Prob}[M(\nu_i(g)) \neq \text{'?'}] \geq \alpha.$$

Similarly, *agent* $i$ *knows a fact* $\psi$ *with confidence* $\alpha$ *relative to a machine* $M$, denoted by

$$g \models_{\mathcal{R}} \mathsf{K}_i^{(\alpha,M)} \psi,$$

iff $g \models_{\mathcal{R}} \mathsf{K}_i^{(\alpha,M)} f_\psi$.

It follows from the above definitions that if $M$ is deterministic, then $g \models_{\mathcal{R}} \mathsf{K}_i^M f$ implies $g \models_{\mathcal{R}} \mathsf{K}_i^{(1.0,M)} f$.

Note that knowledge relative to probabilistic knowledge generators no longer satisfies the positive introspection axiom

$$\mathsf{K}_i \psi \supset \mathsf{K}_i \mathsf{K}_i \psi$$

(when appropriate superscripts are added to the knowledge operators), that is, an agent may not know what it knows. The reason is that determining whether or not $i$ knows $\psi$ with confidence $\alpha$ relative to $M$ depends on the probability $\beta$ with which $M$ outputs values $\neq$ '?'. Determining whether or not the inequality $\beta \geq \alpha$ holds when $\beta$ is very close to $\alpha$ may be computationally difficult. Nevertheless, we do not regard this as a serious deficiency in our approach. Our goal is to reason *about* resource-limited distributed computations; it is not necessary that the logic used for that reasoning itself have an efficient decision algorithm or even be decidable.

## 4 Belief

### 4.1 Implicit Belief

Our goal here is to define the knowledge of an agent participating in a probabilistic system. Consider for example an instrument called an "oilracle" which, once put on the ground, can detect with degree of accuracy .9 whether there is oil underneath. That is, in 90% of the places oilracle gives the correct answer as to whether or not there is oil underground, and in 10% of the places it gives the wrong answer.

Alice is a very fortunate person—1% of the places in her enormous yard have oil under them. As Alice can not dig up the whole yard in search of oil, she uses oilracle to decide where to dig. Suppose it says 'yes'. Alice can reason that there is a 1/12th chance that there is oil in that place. This, however, is technically incorrect, for the presence or absence of oil in a place is not a random event; either there is or there is not oil there, and usage of oilracle can do nothing to change that fact. Rather, the probabilistic statement is really about Alice's chance of finding oil in her yard: Alice knows that the probability is 1/12 that a randomly chosen place has oil, given that oilracle says 'yes'.

Formally, let $\mathcal{G} = \mathcal{G}_{\mathcal{R}}$ be the set of global states of a probabilistic system $\mathcal{R}$, and let $g \in \mathcal{G}$. We say that in state $g$, agent $i$ *implicitly believes* a function $f$ with a degree of confidence at least $\alpha$, denoted by

$$g \models_{\mathcal{R}} \mathsf{B}_i^{\alpha} f,$$

if there exists a value $y$ such that

$$\mathrm{Prob}[y \in f(g') \mid g' \sim_i g] \geq \alpha.$$

Here $g'$ is a random state chosen from the equivalence class $[g]_{\sim_i}$ according to the induced probability distribution.

Putting the example in our formalism, before oilracle has been run, each place in Alice's yard is in one of four possible global states, depending on whether or not there is oil underneath and whether or not oilracle would say 'yes' if put in that place. Alice cannot distinguish between these four states, so for each of these states $g$,

$$g \models \mathsf{B}_{\text{Alice}}^{1/100}(\text{there is oil}) \wedge \mathsf{B}_{\text{Alice}}^{99/100}(\text{there is no oil}).$$

After interrogating oilracle, Alice can distinguish those states where it said 'yes' from those where it said 'no', so for each state $g'$ in which it said 'yes',

$$g' \models \mathsf{B}_{\text{Alice}}^{1/12}(\text{there is oil}) \wedge \mathsf{B}_{\text{Alice}}^{11/12}(\text{there is no oil}).$$

### 4.2 Relative Belief

Relative belief is the feasible version of belief, i.e., the probabilistic version of relative knowledge.

A "knowledge generator" that is allowed to err is called a "belief generator". Intuitively, an $i$-feasible belief generator $M$ for a function $f$ is an oracle which, for every global state $g$, takes $\nu_i(g)$ as input. It may or may not give information about a value of $f(g)$, and when it does, the information may or may not be correct. Thus, any probabilistic polynomial time

Turing machine can be considered to be an $i$-feasible belief generator; presumably, the less often it gives incorrect information the more useful it is. We say that $M$ *lies in* $g$ (for a particular computation) if the output $M(\nu_i(g))$ in that computation is not in $f(g) \cup \{\text{'?'}\}$.

Given a probabilistic system $\mathcal{R}$ whose set of global states is $\mathcal{G} = \mathcal{G}_{\mathcal{R}}$, let $\approx_{\mathcal{R}}$ be an equivalence relation on $\mathcal{G}$, which we call the *context*, and let $\alpha \in [0,1]$. Both $\approx_{\mathcal{R}}$ and $\alpha$ are parameters that are used to specify the reliability of the belief generator. Let $i$ be an agent in the system, let $f$ be a function with domain $\mathcal{G}$, and let $M$ be a probabilistic polynomial time Turing machine. $M$ is an *$i$-feasible $(\alpha, \approx_{\mathcal{R}})$-reliable belief generator for $f$* if for every $g \in \mathcal{G}$, the probability is at least $\alpha$ that $M$ does not lie on a random $g'$ which is chosen from $[g]_{\approx_{\mathcal{R}}}$ according to the underlying probability distribution on global states. In other words, $M$ has to give good answers (i.e., a correct value of $f$ or '?') at least $\alpha$ of the time within each equivalence class of $\approx_{\mathcal{R}}$. Thus, the finer the equivalence relation and the larger the value of $\alpha$, the stronger this restriction becomes.

Let $g \in \mathcal{G}$ be a global state. We say that an agent $i$ *believes $f$ with confidence at least $\alpha$ in context $\approx_{\mathcal{R}}$ relative to a machine $M$*, denoted by

$$g \models_{\mathcal{R}} \mathsf{B}_i^{(\alpha, \approx_{\mathcal{R}}, M)} f,$$

iff $M$ is an $i$-feasible $(\alpha, \approx_{\mathcal{R}})$-reliable belief generator for $f$, and

$$\mathrm{Prob}[M(\nu_i(g)) \in f(g)] \geq \alpha.^3$$

Unlike the case of relative knowledge, relative belief does *not* imply implicit belief, i.e., $g \models_{\mathcal{R}} \mathsf{B}_i^{(\alpha, \approx_{\mathcal{R}}, M)} \psi \not\Rightarrow g \models_{\mathcal{R}} \mathsf{B}_i^{\alpha} \psi$. This corresponds to our intuition: Implicit belief assumes no resource boundedness; relative belief is belief given bounded resources, i.e., partial information. It is conceivable that one would believe less if one were given more information (e.g., have implicit belief). In the cases where the $\approx_{\mathcal{R}}$ relation is a refinement of the $\sim_i$ relations, relative belief does imply implicit belief.

## 5 Interactive Proofs

One of our goals in developing the concepts of relative knowledge and relative belief has been to explain the knowledge that is conveyed in an interactive proof [GMR85]. Consider for example the simple zero-knowledge interactive proof shown in Figure 1, which describes two protocols, $P$ and $V$, to be run by two agents, a prover $p$ and a verifier $v$. The goal of the protocols is for $p$ to convince $v$ that $p$ knows $y$ to be a quadratic residue (perfect square) modulo $n$.

Intuitively, if $p$ cannot efficiently compute a square root of $y$ modulo $n$ and yet $v$ accepts, then $v$ must have received a $w_b$ that satisfies the test in step 4 of $V$. However, since $p$ doesn't know a square root of $y$, it must be the case that $p$ cannot compute a $w_0$ and a $w_1$ that both satisfy $v$'s test. (If $p$ could compute both, then $p$ could compute $w_1 w_0^{-1} \bmod n$, which is a square root of $y$ modulo $n$.) Hence, $p$ must have been able to produce at most one of $w_0$ and $w_1$, and just by chance, that is the one at each stage that $v$ requested in step 2. The probability of this happening is only $1/2$, hence, the probability that $v$ accepts and $p$

[3] We remark that the same $\alpha$ is used to bound both the probability of lying and the probability that a non-'?' answer is produced.

| Protocol $P$ | Protocol $V$ |
|---|---|
| 1. Generate a random $z$; Send $u = z^2 \bmod n$ to $v$. | 1. Wait until $u$ is received. |
| 2. Wait until $b$ is received. | 2. Generate a random $b \in \{0,1\}$ (with equal probabilities); Send $b$ to $p$. |
| 3. Send $w_b = zx^b \bmod n$ to $v$ ($x$ is a fixed square root of $y$.) | 3. Wait until $w_b$ is received. |
| | 4. If $w_b^2 = uy^b \pmod{n}$ then accept else reject. |

Figure 1: Zero-knowledge proof of quadratic residuosity.

cannot efficiently compute a square root of $y$ is at most $1/2$. Similarly, if we let $v$ run $V$ for $t$ itreations and accept only if it accepted in all $t$ iterations, then the probability that $v$ accepts and $p$ cannot efficiently compute a square root of $y$ is at most $1/2^t$.

We want to capture the intuition described above in our formalism. We therefore assume that $v$ is indeed following $V$, and that $p$ is following some arbitrary protocol $Q$. Let $(y, n) \in \mathrm{N} \times \mathrm{N}$ denote some shared input of the system. Let $\sqrt{y}$ be a function that for every $g \in \mathcal{G}$ returns all the square roots of $y(g)$ modulo $n(g)$. We can show the following:

**Theorem 1** *Let $g \approx g'$ iff $n(g) = n(g')$, $y(g) = y(g')$, and $g$ and $g'$ both result from the same number of iterations of the protocol. Then there exists a $v$-feasible belief generator $M_V$ and a $p$-feasible knowledge generator $M_Q$ such that, for every global state $g$ in which $v$ is in an accepting state after running the protocol for $t$ iterations and for every $\varepsilon \in [0,1]$, if $\delta = 1/(2^t\varepsilon)$, then*

$$g \models \mathsf{B}_v^{(1-\delta,\approx,M_V)}\mathsf{K}_p^{(1-\varepsilon,M_Q)}\sqrt{y}.$$

In words, when $v$ accepts, it believes with confidence $1-\delta$ that $p$ "can compute" $\sqrt{y}$ with probability at least $1-\varepsilon$. Note that $\varepsilon$ is arbitrary, but the smaller it is, the larger $v$'s uncertainty $\delta$ becomes. The two become equal when $\varepsilon = 1/2^{(t/2)}$, so both can simultaneously be made exponentially small in $t$.

We sketch here the main ideas of the proof. Details are deferred to the full paper.

We begin by looking in a little more detail at the interaction of the arbitrary protocol $Q$ with $t$ rounds of the fixed protocol $V$. During such a run, both $p$ and $v$ may toss coins. Let $\pi_p$ and $\pi_v$ be their coin toss sequences, respectively. The run is uniquely determined by $\pi_p$ and $\pi_v$. Call the run *accepting* if $v$ accepts in the end.

We are now in a position to define $M_Q$: $M_Q$ receives as input the local view of $p$ at the end of the protocol. Recall that we assume agents do not forget, so the local view of every proceses contains the complete local history of the run. Moreover, because $v$ sends all of its coin tosses to $p$, $p$'s local view alone contains enough information to completely reconstruct the entire run, for given $p$'s view of a global state $g$, $M_Q$ can construct the coin toss sequences $\pi_p$ and $\pi_v$ that determine the run $r = r_g$, and then simulate $Q$ and $V$ to reconstruct $r$ itself.

Assume that $r$ is accepting. Then $v$ receives a good value in step 3 of its protocol at each iteration, where "good" means that it passes the test in step 4. For each $i$, $1 \leq i \leq t$,

$M_Q$ will carry out a simulation of $Q$ interacting with $V$ using $p$'s coin toss sequence $\pi_p$ and $v$'s coin toss sequence $\pi'_{v,i}$. The latter is identical to $\pi_v$ except that the $i^{\text{th}}$ bit has been flipped. The simulation is carried out for $i$ iterations of $V$. Denote the resulting run by $r'_i$. Because $\pi'_{v,i}$ and $\pi_v$ agree in the first $i-1$ positions, $r'_i$ and $r$ are identical through the first $i-1$ iterations. Let $b$ and $b'$ be the $i^{\text{th}}$ bits of $\pi_v$ and $\pi'_{v,i}$, respectively, and let $w_b$ and $w_{b'}$ be the corresponding values received by $v$ in step 3 of the $i^{\text{th}}$ iteration. By construction, $b \neq b'$. We know that $v$ accepts $w_b$ in step 4. If $v$ also accepts $w_{b'}$, then either $w_b/w_{b'}$ or its inverse is a square root of $y$, so $M_Q$ can determine which and output it. Consequently, if for some $i$, $1 \leq i \leq t$, $r'_i$ is accepting, then $M_Q$ outputs a correct $\sqrt{y}$. If $r'_i$ is not an accepting run for any $i$, then $M_Q$ outputs '?'. Also, $M_Q$ outputs '?' in case $r$ itself is not an accepting run.

In computing $r'_i$, it may be the case that $p$ needs more coin tosses than are contained in $\pi_p$. If so, then $M_Q$ extends $\pi_p$ as necessary by flipping coins itself. Our analysis assumes that the same $\pi_p$ is used for each of the $t$ simulations; hence, whenever $\pi_p$ is extended during one simulation, the extended version is used in subsequent iterations.

Let $\alpha_Q$ be the probability that a random run of $Q$ with $V$ accepts. We argue that the probability that $M_Q$ produces a square root of $y$ on a random run is at least $\alpha_Q - 1/2^t$. Consider a fixed, sufficiently long prover's coin toss sequence $\pi_p$, and consider the $2^t$ runs obtained for each of the $2^t$ verifier coin toss sequences of length $t$. If two or more of those runs are accepting, then, given any of those runs, $M_Q$ will always succeed. Hence, the only accepting runs on which $M_Q$ fails to produce a square root of $y$ are those in which all of the other $2^t - 1$ related runs (with the same prover's coin tosses) are non-accepting. The probability of this occurrence is at most $1/2^t$. Note that if $\pi_p$ is not sufficiently long, then $M_Q$ extends it randomly, and the above remarks apply to the resulting extension.

Now, let $\gamma$ be the probability that the formula

$$g \models \mathsf{K}_p^{(1-\varepsilon, M_Q)} \sqrt{y}$$

holds in a random accepting state $g$. We can show that

$$\gamma \geq 1 - \frac{1}{\alpha_Q 2^t \varepsilon}.$$

Let $M_V$ be the simple belief generator that outputs **true** on all accepting states after $t$ rounds and outputs '?' elsewhere. The probability that $M_V$ lies on a random run $r$ is at most $(1-\gamma)\alpha_Q$, for $M_V$ only lies when $r$ is accepting and $\neg\mathsf{K}_p^{(1-\varepsilon, M_Q)}\sqrt{y}$ holds. Therefore, the probability that $M_V$ does not lie is

$$1 - (1-\gamma)\alpha_Q \leq 1 - \frac{1}{2^t \varepsilon} = 1 - \delta.$$

The theorem follows.

## 6 Cryptographic Protocols

*Cryptographic protocols* extend distributed protocols by being non-deterministic as well as probabilistic. Each agent is allowed at the beginning to non-deterministically choose a protocol from some set of possible probabilistic protocols. The set of chosen protocols forms a system which is then used to obtain a random run.

More precisely, each agent $i$ has a set of possible probabilistic protocols, denoted by $\mathcal{P}_i$. Initially, each agent chooses (non-deterministically) a protocol $P_i \in \mathcal{P}_i$. This choice defines a system $\mathcal{R}$ over a set of global states $\mathcal{G}_{\mathcal{R}}$. Thus, a *cryptographic protocol* is a family of (probabilistic) systems.

Because there is no probability distribution on the choices that an individual agent makes, we cannot make probabilistic statements about the outcome of a run of a system chosen in this way. Rather, the only statements of interest are those that are valid for all possible systems allowed by the protocol.

Looking back at Theorem 1, we see that the particular protocol chosen by the prover enters into the statement of the theorem at one place—namely, the knowledge generator $M_Q$ depends on $Q$, the protocol run by $p$. In order to get a statement valid about cryptographic protocols instead of just systems, we must modify our definitions slightly.

First of all, we interpret formulas at pairs $(\mathcal{R}, g)$ where $\mathcal{R}$ is the system (i.e., the protocols chosen by each of the individual agents) and $g \in \mathcal{G}_{\mathcal{R}}$ is a global state. For example, in an interactive proof system, the formulas are interpreted over pairs of the form $(\mathcal{R}, g)$, where $\mathcal{R} \in \mathcal{P}_p \times \mathcal{P}_v$ and $g \in \mathcal{G}_{\mathcal{R}}$. Secondly, we replace the knowledge (resp. belief) generator in the superscripts of the $\mathsf{K}$ (resp. $\mathsf{B}$) operator with a family of knowledge (resp. belief) generators, indexed by the protocol being run by the agent $i$ associated with the operator. Thus, $\mathsf{K}_i^{M_{P_i}}$ is replaced by $\mathsf{K}_i^{\mathcal{M}}$, where $\mathcal{M}$ maps the protocol $P_i$ run by agent $i$ to a knowledge generator $M_{P_i}$. The interpretation of, say, $((Q, V), g) \models \mathsf{K}_p^{\mathcal{M}}\psi$ is the same as $g \models \mathsf{K}_p^{M_Q}\psi$, where $M_Q = \mathcal{M}(Q)$ is the knowledge generator corresponding to the protocol $Q \in \mathcal{P}_p$. The $\mathsf{B}$ operator is handled similarly.

In this way, we get the following restatement of Theorem 1:

**Theorem 2** *Let $\mathcal{P}$ be a family of prover protocols and $V$ the verifier protocol of Figure 1. For each $Q \in \mathcal{P}$ let $\mathcal{R}_Q$ denote the system $(Q, V)$ and define $\approx_{\mathcal{R}_Q}$ such that $g \approx g'$ iff $n(g) = n(g')$, $y(g) = y(g')$, and $g$ and $g'$ both result from the same number of iterations of the protocol. Then there exists a family of knowledge generators $\mathcal{M}$ and a single belief generator $M_V$ such that for all probabilistic polynomial time prover protocols $Q \in \mathcal{P}$, all $(y, n) \in \mathbf{N} \times \mathbf{N}$ and all $\varepsilon \in [0, 1]$, if $\delta = 1/(2^t\varepsilon)$ and $g$ is a global state in which $V$ accepts after running the verifier protocol for $t$ iterations, then*

$$(\mathcal{R}_Q, g) \models \mathsf{B}_v^{(1-\delta, \approx_{\mathcal{R}_q}, M_V)} \mathsf{K}_p^{(1-\varepsilon, \mathcal{M})} \sqrt{y}.$$

Details of the proof are deferred to the full paper.

## 7 Current Research and Open Problems

We presented a logic of relative knowledge and belief that enables us to reason about computable knowledge in deterministic as well as in probabilistic systems. This logic has numerous applications, especially in the area of cryptographic protocols. We demonstrated one application by defining what it is that the verifier learns when it accepts after running an interactive proof of knowledge of a square root in $\mathbf{Z}_n^*$. We are currently trying to apply the ideas presented in this paper to formalize what it is that the verifier does *not* learn in order to make formal sense of the notion of "knowledge" in zero-knowledge proofs.

## Acknowledgements

We would like to thank Martin Abadi, Jin-Yi Cai, Merrick Furst, Neil Immerman, Silvio Micali, Yoram Moses, and Martin Tompa for helpful and stimulating discussions.

## References

[CM86] K. M. Chandy and J. Misra, How processes learn, *Distributed Computing* 1:1, 1986, pp. 40–52.

[FFS87] U. Feige, A. Fiat, and A. Shamir, Zero knowledge proofs of identity, *Proc. 19th ACM Symp. on Theory of Computing*, 1987, pp. 210–217.

[FH85] R. Fagin and J. Y. Halpern, Belief, awareness, and limited reasoning, *Proc. of the 9th IJCAI*, 1985, pp. 491–501. Revised version to appear in *Artificial Intelligence.*

[FI86] M. J. Fischer and N. Immerman, Foundations of knowledge for distributed systems, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference* (J. Y. Halpern, ed.), Morgan Kaufmann, 1986, pp. 171–186.

[GHY85] Z. Galil, S. Haber, and M. Yung, A private interactive test of a Boolean predicate and minimum knowledge public key cryptosystems, *Proc. 26th IEEE Symp. on Foundations of Computer Science*, 1985, pp. 360–371.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof-systems, *Proc. 17th ACM Symp. on Theory of Computing*, 1985, pp. 291–304.

[Hal86] J. Y. Halpern, Reasoning about knowledge: an overview, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference* (J. Y. Halpern, ed.), pp. 1–17, Morgan Kaufmann, 1986.

[Hal87] J. Y. Halpern, Talk at evening rump session, 6th ACM Symp. on Principles of Distributed Computing, August 1987.

[HF85] J. Y. Halpern and R. Fagin, A formal model of knowledge, action, and communication in distributed systems: preliminary report, *Proc. 4th ACM Symp. on Principles of Distributed Computing*, 1985, pp. 224–236.

[HM84] J. Y. Halpern and Y. Moses, Knowledge and common knowledge in a distributed environment, *Proc. 3rd ACM Symp. on Principles of Distributed Computing*, 1984, pp. 50–61. A revised version appears as *IBM Research Report RJ 4421*, Aug., 1987.

[HZ87] J. Y. Halpern and L. D. Zuck, A little knowledge goes a long way: simple knowledge-based derviations and correctness proofs for a family of protocols, *Proc. 6th ACM Symp. on Principles of Distributed Computing*, 1987, pp. 269–280.

[Mos87] Y. Moses, Resource-bounded knowledge and belief, unpublished manuscript, 1987. To appear as a Weizmann Institute Technical Report.

[TW87] M. Tompa and H. Woll, Random self-reducibility and zero knowledge interactive proofs of possession of information, *Proc. 28th IEEE Symp. on Foundations of Computer Science*, 1987, pp. 472–482.